

Воронина Наталья (Национальный каталог)



А зачем?

Проблемы

1. Увеличение количества потребителей
2. Медленная скорость обработки запросов
3. Желание команды писать на Golang

Варианты

1. Снизить количество запросов
2. Оптимизировать код, оптимизировать запросы в базу, добавить кэширование
3. Смириться с потерей части команды



RoadRunner — что это за зверь?



RoadRunner — это высокопроизводительный сервер приложений PHP, балансировщик нагрузки и диспетчер процессов, написанный на Golang.







А какие альтернативы?

PHP-FPM, PHP-PM, Nginx-Unit, React-PHP и RoadRunner

RoadRunner is one of the fastest ways to run your PHP application

0
1000
2000
3000

Response Time, ms



«Классическое» PHP-приложение 2022

Server

Kubernetes

Client

Load Balancer

nginx

PHP-fpm

PHP-fpm

PHP-fpm

PHP Code

PHP Code

PHP Code

ClickHouse

DB

Redis/ Memcache/ Elasticsearch …

Kafka/RabbitMQ



Roadrunner, встроенный в архитектуру

Server

Client

Load Balancer

Kubernetes

nginx

RoadRunner

Worker pool

Service

RPC

PHP Worker

PHP Worker

PHP Worker

ClickHouse

DB

Redis/ Memcache/ Elasticsearch …

Kafka/RabbitMQ



Плюсы и минусы использования

Плюсы

- Ускорение времени ответа
- Возможность построения гибридных PHP-Go-приложений
- Возможность предобработки запроса на Go
- Увеличение RPS, которое держит приложение

Минусы

- Необходимость адаптации кода
- Возможны утечки памяти
- Сложность отладки
- Высокие риски при выкладке на production



А что, если у меня есть фреймворк?

Тогда вам повезло, скорее всего, всю работу за вас кто-то уже сделал. Готовые «мосты» для перехода есть под:

CakePHP
Laravel
Slim
Spiral Framework
Symfony Framework
Symlex Framework
Ubiquity Framework
Zend Expressive
Yii2 and Yii3
Phalcon3 and Phalcon4
Mezzio
Chubbyphp Framework
CodeIgniter













Решились на переход!





Что обновляем

- PHP 8+ (возможно, на более ранних версиях тоже можно, но в нашем случае уже начали переход на 8.1, поэтому дождались завершения перехода)

- Psr7 Request/Response.
  - Брался request из php://input.
  - Файлы грузились из $_FILES
  - Использовался setcookie() для установки сессии, пришлось переписывать на:

```
$response->withAddedHeader('Set-Cookie', $value);
```




Чистка состояния

- Убираем любые операции в __destruct()

- Чистим статику

Пример:

```
protected function clearState(): void
{
    Sessions::clearInstances();
    Database::clearConnections();
    ProductsSendMessageService::sendAllToQueue();
}
```




Отладка




```
var_dump($value);
print_r($value);
die;
echo $value;
exit;
dd();
```







Локальное окружение

- Переписываем Dockerfile с php-fpm и nginx
- Переходим на xdebug
- Настраиваем rr.yaml
- Устанавливаем с Packagist зависимости:

* "spiral/roadrunner": "v2.11.1",
"spiral/roadrunner-cli": "v2.1.0",
"spiral/roadrunner-http": "v2.0.4",
"spiral/roadrunner-worker": "v2.1.5"



Nginx

Если необходимо запустить через nginx:

```
location / {
   proxy_pass http://php:8083;
}
```

Что это дает?

- Не нагружаем RR запросами на статику или политикой cors.
- Минимизируем риски при первом переходе



DEV-конфигурации файла rr.yaml

```yaml
version: '2.7'
rpc:
  listen: tcp://127.0.0.1:6003
server:
  command: "php index.php"
  env:
    - XDEBUG_SESSION: 1
    - PHP_IDE_CONFIG: "serverName=nk_platform"
```

version – это версия конфигурации. Не самого RR!

rpc – используется для подключения к RR из PHP-воркеров. Например, если не указать, то и лог RR собирать не сможет

server:
  command – адрес, по которому будут доступны ваши PHP
  env: переменные окружения, содержащие конфиги для Xdebug



DEV-конфигурации файла rr.yaml

```yaml
http:
  address: "0.0.0.0:8083"
  pool.debug: true
#  pool.num_workers: 2
logs:
  mode: development
  level: warn
  file_logger_options:
    log_output: '../logs/rr_api.log'
```

http:
address – адрес, по которому будет доступно ваше PHP-приложение
pool.debug: true – означает, что будет запущен с перезагрузкой под каждый запрос. В режиме разработки является оптимальной настройкой, так как вам не надо будет перезапускать сервер для обновления кода
pool.debug: false + **pool.num_workers**: 2 означает, что будет запущен один воркер

logs – блок отвечает за вывод логов и уровень логирования

PROD-конфигурации файла rr.yaml

```yaml
http.pool:
    num_workers: 4
    max_jobs: 0
    allocate_timeout: 60s
    destroy_timeout: 60s
    debug: false
    supervisor:
      watch_tick: 1s
      ttl: 180s
      max_worker_memory: 1850
      exec_ttl: 0s
```

http.pool:

num_workers: n – количество запускаемых воркеров. Зависит от ваших доступных логических ядер

max_jobs: 0 – максимальное количество итераций в одном воркере. 0 – без ограничений

allocate_timeout – время в мс на запуск воркера

destroy_timeout – время в мс на уничтожение воркера

pool.supervisor

watch_tick – проверка, что воркер жив

ttl – максимальное время жизни воркера

max_worker_memory – максимальная потребляемая память воркером в мегабайтах

exec_ttl: 0 – максимальное время, данное воркеру на выполнение запроса. 0 – без ограничений



PROD-конфигурации файла rr.yaml

```yaml
metrics:
  address: "0.0.0.0:8085"
status:
  address: "0.0.0.0:2118"
```

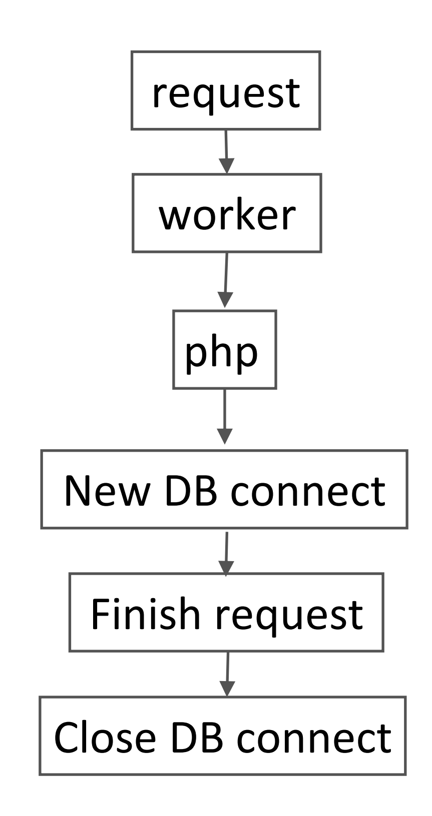
metrics – адрес сбора метрик в Prometheus
status – healthcheck для мониторинга системы



Коннекты к базе данных

Singleton-коннект к БД, который хранит его в статике.

Если нас устраивает установка соединения на каждый запрос, то не забываем чистить статику.



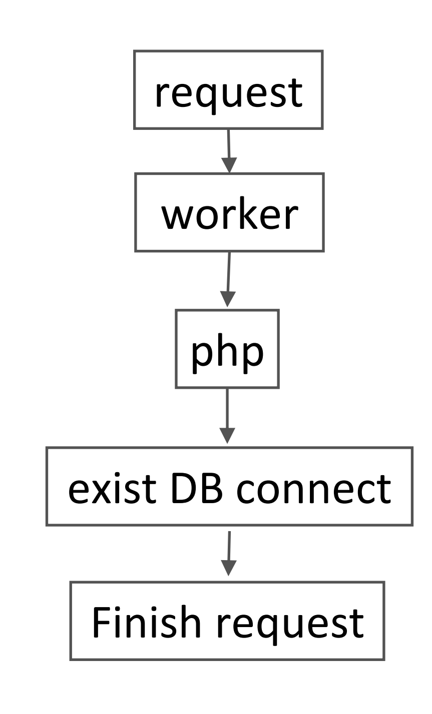
Пример прохода одного запроса



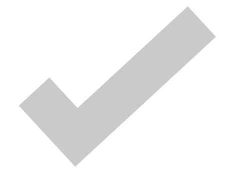
Хранение коннектов к базе данных

- Singleton-коннект к БД оставляем
- НЕ чистим статику
- Реализуем переподключение к БД в случаях потери соединения
- Не забываем: сколько воркеров, столько соединений. БД-параметр

max_connections = workers rr + cronjobs + queue listeners * n



Пример прохода одного запроса



Перестаем тратить время на установку соединения



Чего удалось достичь?



А какие проблемы?

ALLURE REPORT 9/15/2022
15:54:31 - 16:18:24 (23m 52s)

1016
test cases

35.33%

После выкладки на тест

2 недели

ALLURE REPORT 10/4/2022
17:48:58 - 18:18:55 (29m 56s)

1017
test cases

99.21%

К релизу



Локальный прогон тестов

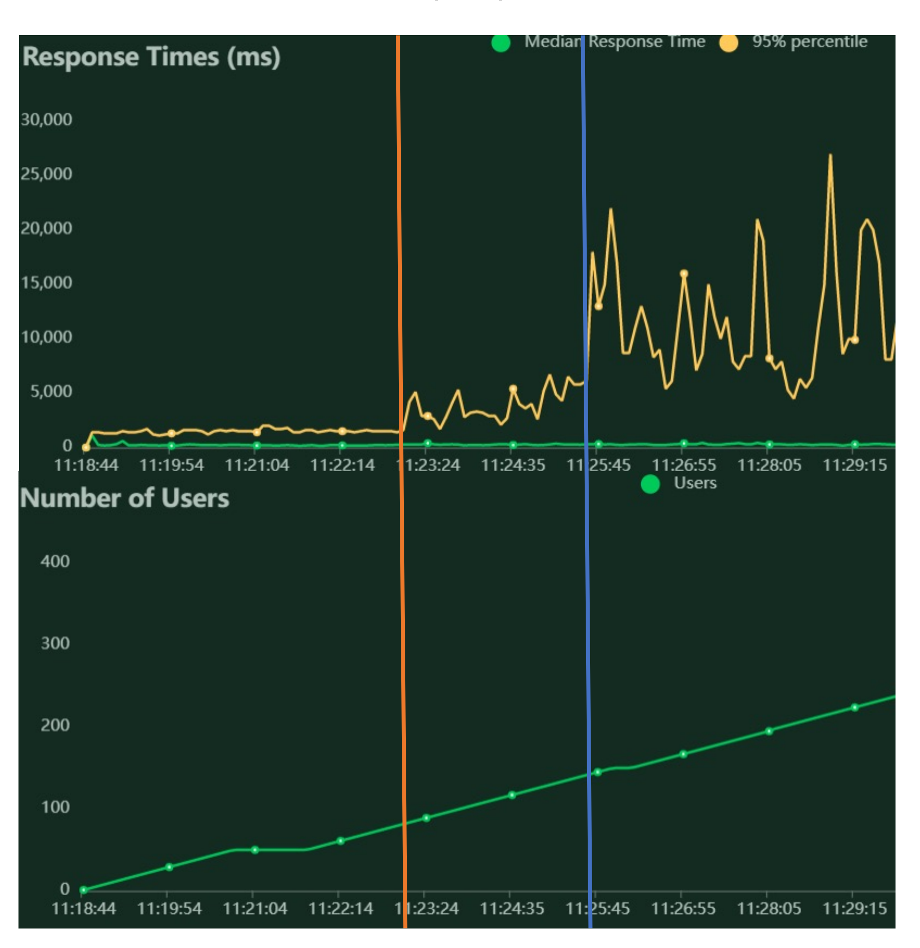
Страница	php-fpm	roadrunner with reload	roadrunner without reload
Список модераций	1.04 s / 486 ms	1.29 s / 702 ms	365 ms / 320 ms
Главная страница	469 ms / 390 ms	445 ms / 384 ms	65 ms/ 44 ms
Список аккаунтов	4.99 s / 1.03 s	1.76 s / 936 ms	567 ms / 552 ms
Товар по id	955 ms / 445 ms	961 ms / 724 ms	153 ms / 142 ms

По результатам анализа эффективности roadrunner (в таблице приведены первый запрос / повторный запрос)



Прогон на тестовом стенде

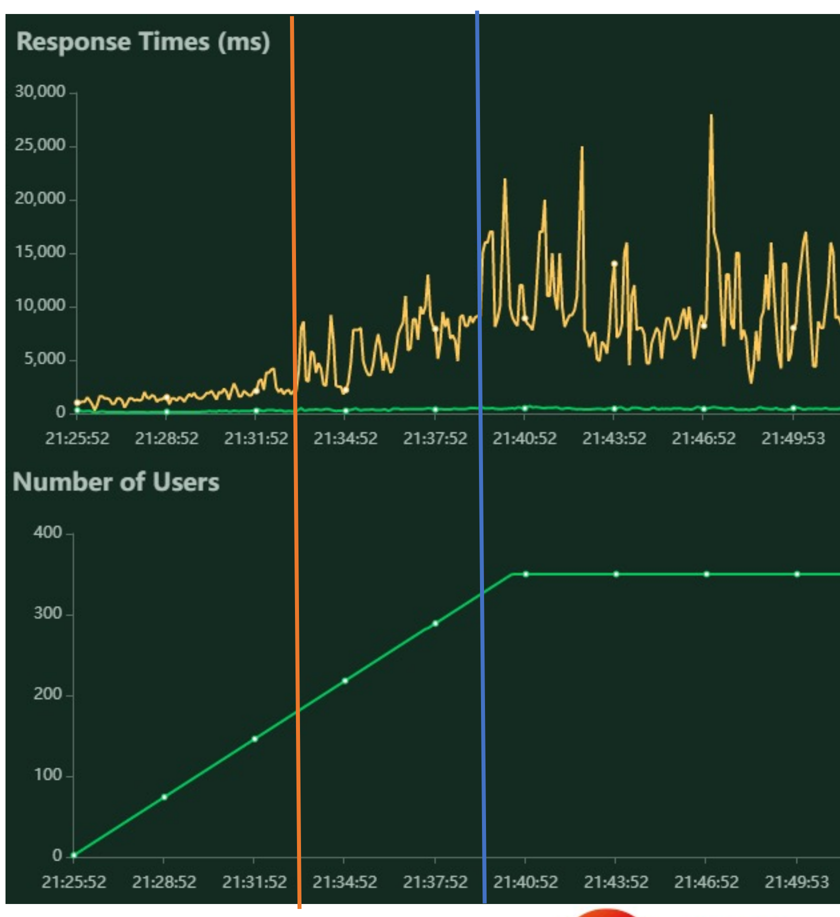
Php-fpm



Первые задержки ответа:
75u -> 180u

Существенные замедления ответа:
120u -> 320u

RoadRunner





Системные проблемы

Pods memory usage

1.30 GiB
1.21 GiB
1.12 GiB
1.02 GiB
954 MiB
858 MiB
763 MiB
668 MiB
572 MiB
477 MiB
381 MiB
286 MiB
191 MiB
95.4 MiB
0 B

10/11 16:00 10/11 20:00 10/12 00:00 10/12 04:00 10/12 08:00 10/12 12:00 10/12 16:00 10/12 20:00 10/13 00:00 10/13 04:00 10/13 08:00 10/13 12:0

CPU – без изменений

Сеть – без изменений

Задействованная оперативная память – выросла (график)



Итоги

Положительные стороны перехода

- ✓ Удалось достигнуть пропускной способности в 3 раза выше
- ✓ Уменьшилось время ответа в 2-8 раз
- ✓ Появилась возможность реализовать гибридное PHP-Go-приложение

Почему переход не всегда оправдан

- ❑ Переписали много кода ядра (базовый diff на 10000+ строк)
- ❑ Пол года на переход
- ❑ Есть проблемы приложения в работе с памятью
- ❑ Усложнилась разработка и отладка



Если есть желание попробовать



- Пример локального окружения
- Две точки входа (api, public)
- Простейшее API
- Отображение через Twig

https://github.com/NVoronina/rr-playground



Отзывы по докладу



Спасибо за внимание!

Telegram @NVoronina

